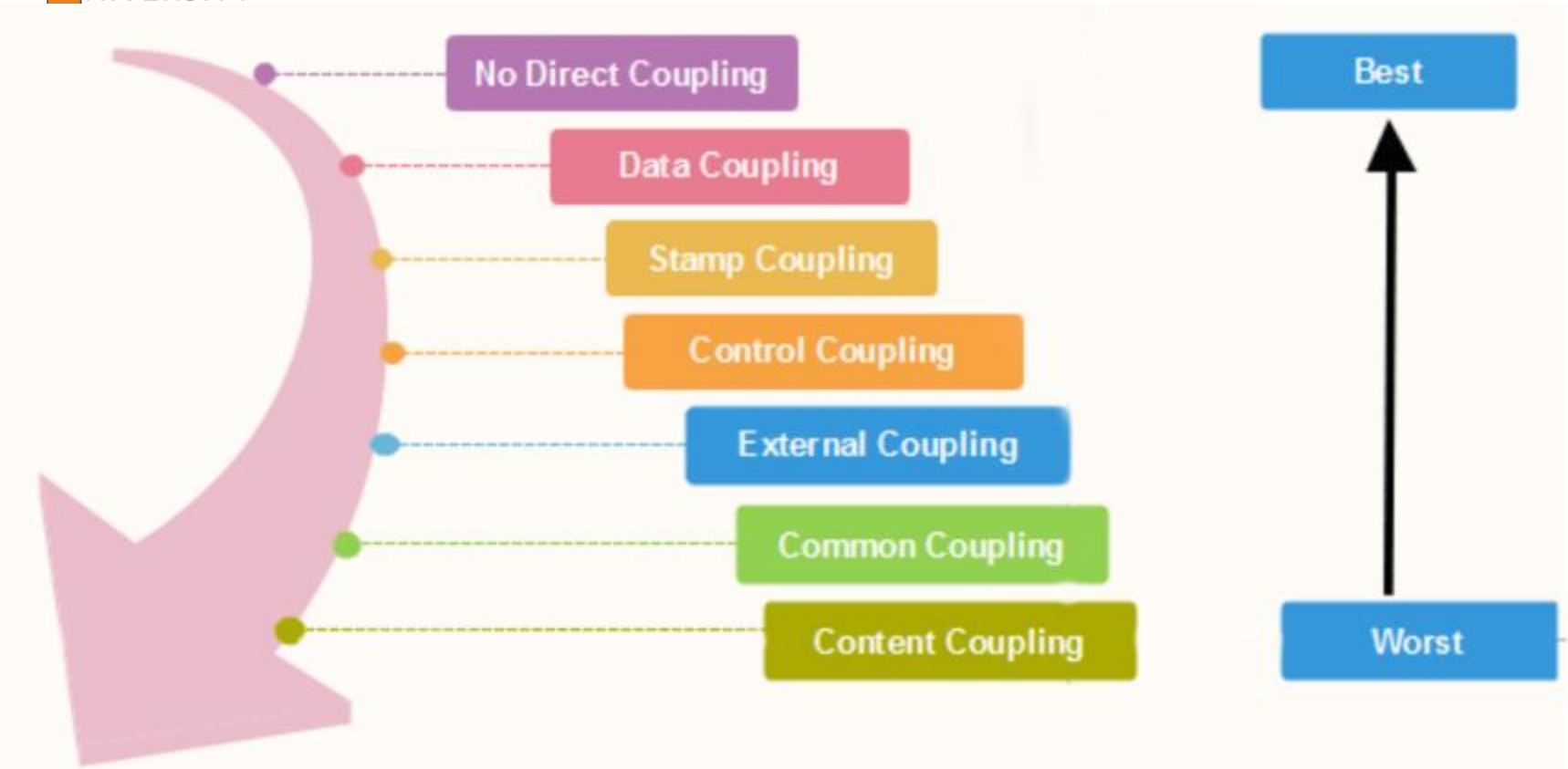
LOVELY PROFESSIONAL UNIVERSITY

NAAC GRADE A++

Unit-2

S/w Design


No Direct Coupling
Data Coupling
Stamp Coupling
Control Coupling
External Coupling
Common Coupling
Content Coupling
Best
Worst

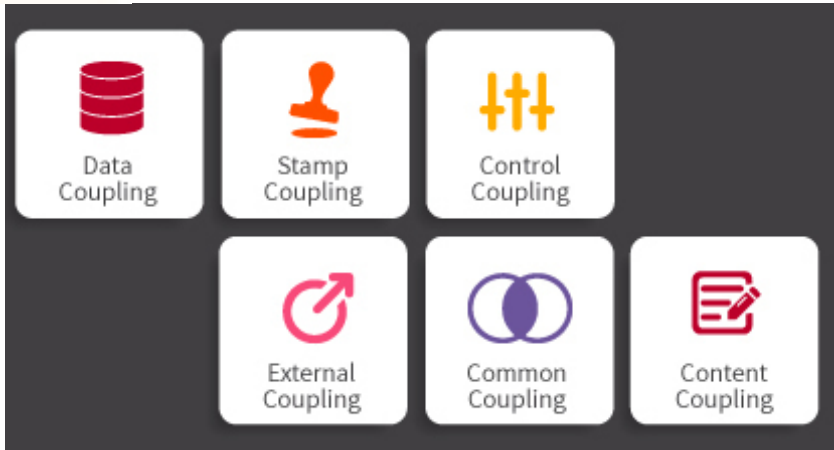
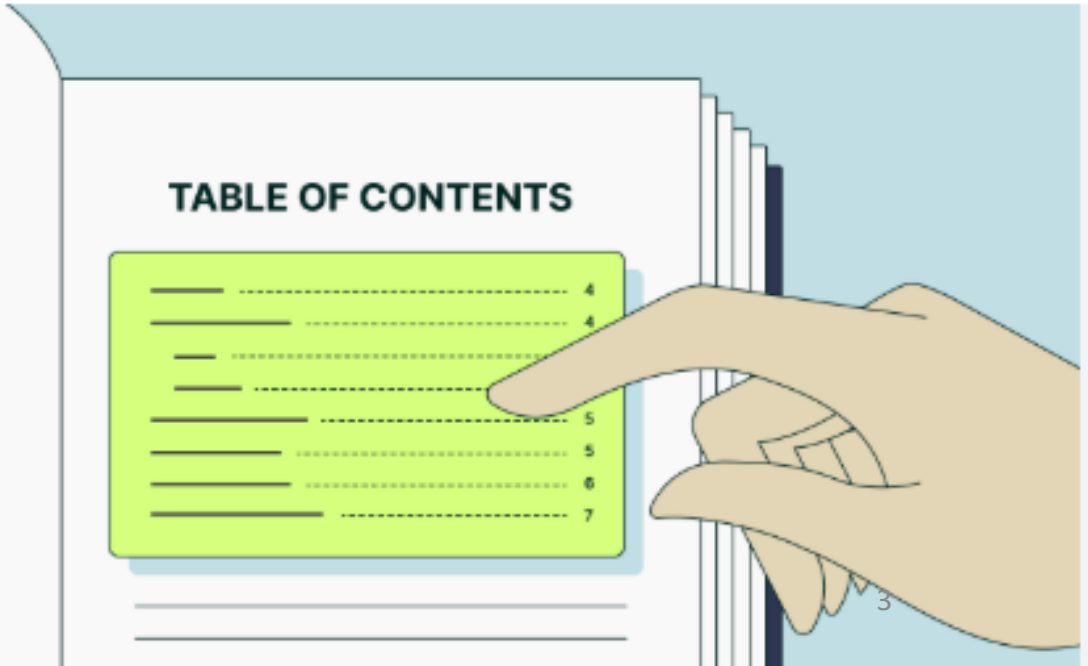
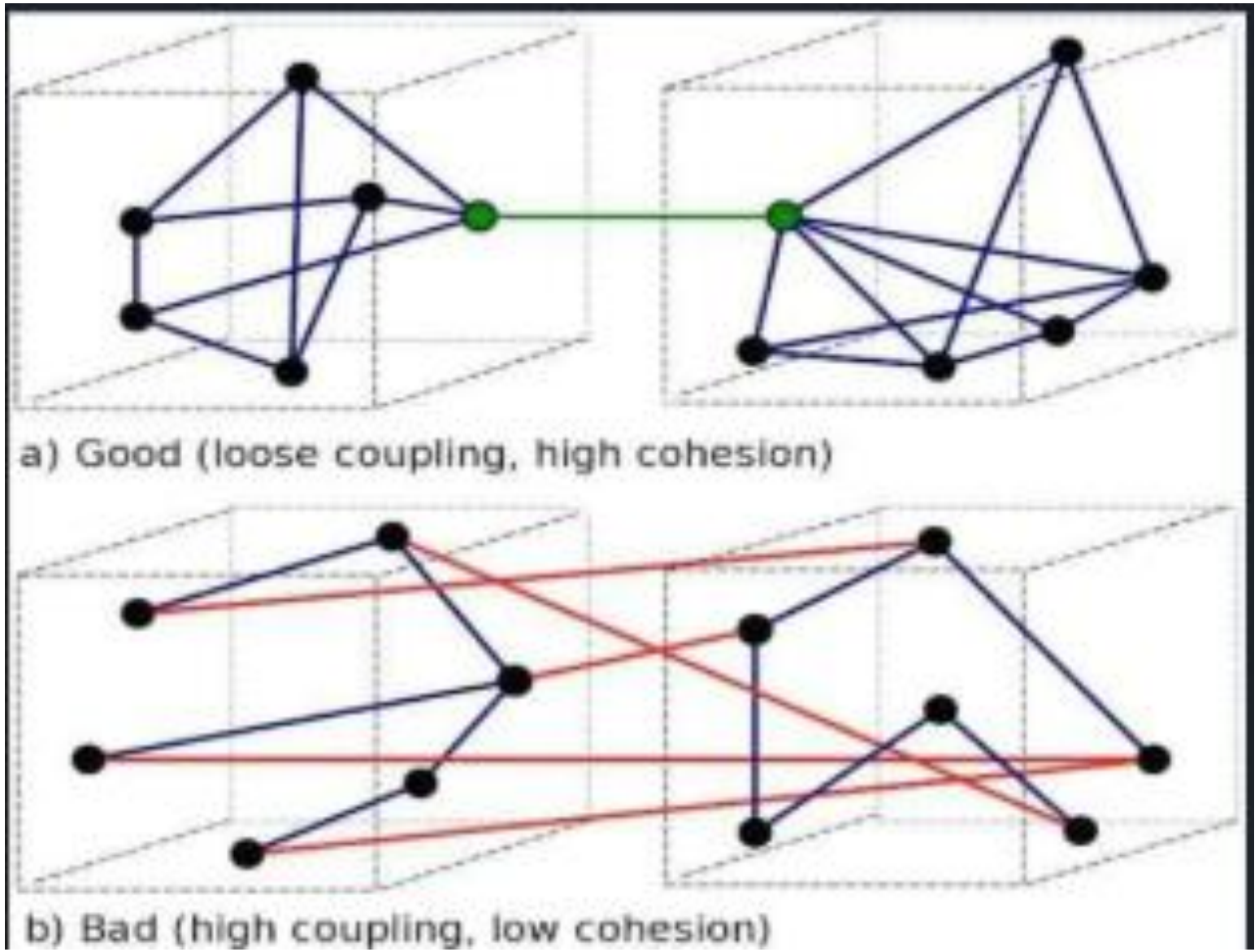

Data Coupling
Stamp Coupling
Control Coupling
External Coupling
Common Coupling
Content Coupling

Table of Content

- Cohesion
- Types of cohesion




a) Good (loose coupling, high cohesion)
b) Bad (high coupling, low cohesion)

What is Cohesion?

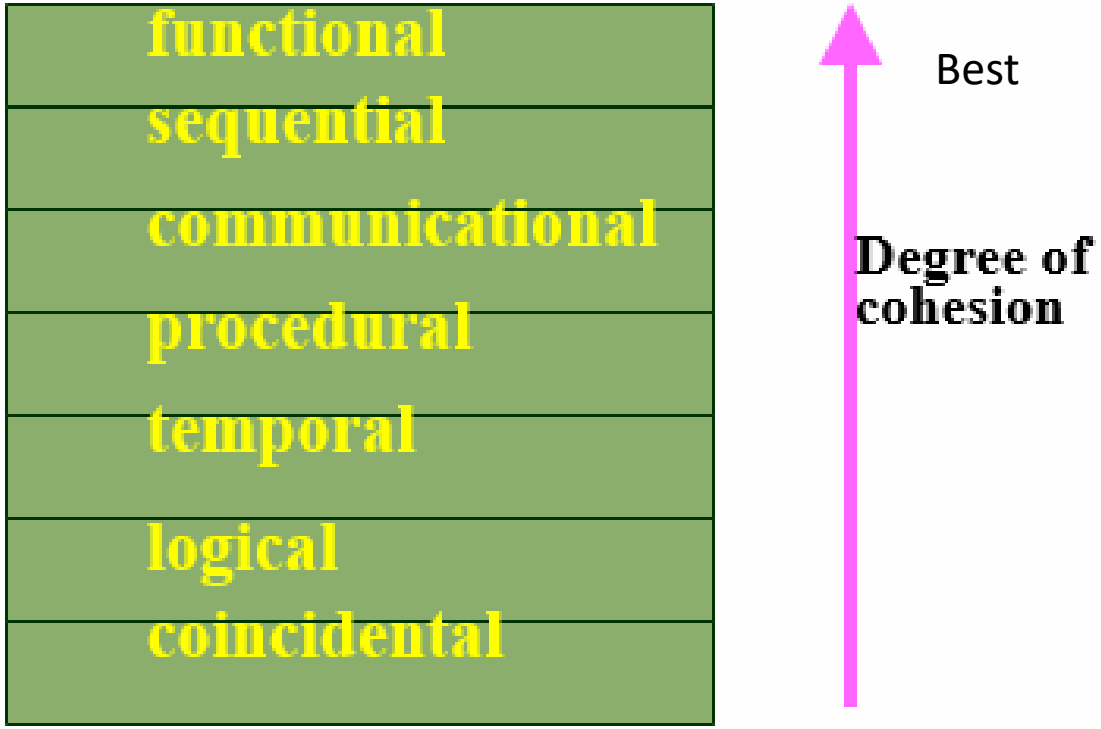
- **Cohesion refers to** the measure of how closely the components within a module or modules itself are related to one another
- High cohesion implies that element within module are strongly related and contribute to perform single well defined task
- **LOW COHESION** suggest that the components are loosely related and perform their own separate task.
- **In good software system, Level of cohesion must be high.**

Why the Cohesion is Important?

- It is the one of the standard measurement of high quality software
- Ensures that the s/w system is maintainable, extensible, reusable and more understandable
- High cohesion leads to modular, understandable code, making it easier to debug, modify and scale


Functional Cohesion
Sequential Cohesion
Communicational Cohesion
Procedural Cohesion
Temporal Cohesion
Logical Cohesion
Coincidental Cohesion

Types of cohesion (categorized from high to low)



- **Cohesion (intra module)**

- Cohesion represent the *detail design (within module, how elements are interrelating to each other)*
- Cohesion means togetherness or group within the module.
- How we are grouping the various functions/ various methods inside the module.

Functional Cohesion (best)

- functional cohesion that occurs when the elements within a module perform a single, well-defined task or a function.
- All the elements within the module contribute to achieving the same objective.
- This type of cohesion is considered the most desirable & strongest.

Example: Bank Account Management Module

- Consider a **bank account management system** where the main task is to handle various banking operations.
- A module can be designed to handle all the functions related to **managing a bank account**, such as depositing money, withdrawing money, and checking the balance.
- Each of these functions is a part of the **single purpose** of managing an account and contributes to this main goal.

Sequential Cohesion

- when the elements within a module are arranged in a specific sequence, with the **output of one element serves as the input for the next element.**
- The elements are executed in a step-by-step manner to achieve a particular functionality.

Examples – Online Payment Processing System

- Each function performs a specific task and feeds its result into the next one.

Steps in the Payment Process:

- **Verify Payment Method**
  - Checks if the payment method (credit card, PayPal, etc.) is valid and has sufficient funds.
- **Authorize Payment**
  - Verifies if the user has permission to make the payment (checks user authentication).
- **Process Transaction**
  - Actually processes the payment, transferring funds from the user to the seller.
- **Generate Receipt**
  - After the transaction is successful, a receipt is generated.
- **Send Confirmation Email**
  - Sends the confirmation email to the customer with details of the payment and receipt.

Communicational Cohesion

Two elements operate on the same input data or contribute towards the same output data.

Example:

Employee Management System

- A module that:
  - Retrieves employee salary details.
  - Updates salary records.
  - Generates salary reports.
- All functions **work on the same employee salary data**.

Procedural Cohesion

A module where functions execute **in sequence**, but are only **loosely related**.

Example:

Order Processing System

- A module that:
  - **Validates order details** (checks item availability).
  - **Processes payment** (deducts balance).
  - **Prints a receipt** (generates invoice).

- Steps are performed in order but serve **different purposes**.

Temporal Cohesion

A module where functions are grouped because they execute at the **same time** or **specific event**, but are not functionally related.

Example: User Logout Module

- A **logout module** in a web application performs multiple tasks when a user logs out:
- **Ends user session** (clears authentication tokens).
- **Logs the logout event** (stores timestamp in logs).
- **Saves user preferences** (e.g., last visited page, theme settings).
- **Clears temporary cache** (removes session-related data).
- **Redirects user to the login page**.

❖ These tasks are **unrelated in function** but must happen **at the same time** when a user logs out.

Logical Cohesion

- A module where functions are grouped because they perform **similar types of tasks**, but the actual operation is determined by a **control flag or parameter**.
- **Characteristics**:
  - Functions are **logically related** but **do different tasks**.
  - A **control flag or condition** decides which function is executed.
  - **Lower cohesion** because unrelated functionalities exist in the same module.

Example: Printer Driver

- A **printer driver module** handles multiple functions:
  - **Print document**
  - **Scan document**
  - **Send fax**
- A **control flag (user input)** decides which operation to execute.

Coincidental Cohesion

- Occurs when functions are **randomly grouped together** in the same module **without a meaningful relationship.**

Example:

"Common Functions" Module in an E-commerce App

- Imagine an e-commerce application where developers create a **Common Functions module** that contains **completely unrelated functionalities**:
- **Apply discount codes** (related to pricing)
- **Track customer location** (related to user tracking)
- **Generate order invoices** (related to transactions)
- **Send promotional emails** (related to marketing)

❖**Issue:** These functionalities **belong to different concerns** but are placed together arbitrarily.

- **What does cohesion refer to in software engineering?**

a) The degree of interdependence between modules
b) The degree to which the elements inside a module belong together
c) The number of modules in a system
d) The level of abstraction in a module

- **Which type of cohesion is the strongest and most desirable?**

a) Logical cohesion
b) Coincidental cohesion
c) Functional cohesion
d) Temporal cohesion

- **Which of the following is an example of poor cohesion?**

a) A module that performs a single well-defined task
b) A module that has multiple unrelated functions
c) A module that processes a specific type of data
d) A module that follows the single-responsibility principle

- **What is the weakest type of cohesion?**

a) Coincidental cohesion
b) Logical cohesion
c) Communicational cohesion
d) Procedural cohesion

- **In which type of cohesion are elements grouped because they operate on the same data?**

a)Logical cohesion
b) Temporal cohesion
c) Communicational cohesion
d) Functional cohesion

- **Which type of cohesion groups elements that perform similar operations but are activated through a control parameter?**

a) Logical cohesion
b) Temporal cohesion
c) Sequential cohesion
d) Functional cohesion

- **A module with high cohesion is beneficial because...**

a) It makes the code more complex
b) It increases module dependencies
c) It enhances code maintainability and reusability
d) It reduces modularization

- **Which type of cohesion is considered better than temporal cohesion but worse than functional cohesion?**

a) Sequential cohesion
b) Coincidental cohesion
c) Logical cohesion
d) Communicational cohesion